Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Janez Udovič

Sistem za preverjanje napak na lesu

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Franc Solina

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Janez Udovič sem avtor diplomskega dela z naslovom:

Sistem za preverjanje napak na lesu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Franca Soline,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 25. marca 2015 Podpis avtorja:

Za vodenje in nasvete se zahvaljujem mentorju na fakulteti prof. dr. Francu Solini. Za strokovno pomoč in vsestransko podporo pa gre posebna zahvala gospodu Miroslavu Barutu, ki mi je omogočil izdelavo prototipa sistema v svojem podjetju MBvision. Zahvaljujem se prijatelju Tinetu in življenjski sopotnici Nataši, ki sta mi s strokovnimi nasveti in vzpodbudami olajšala prenekatero zagato med študijem in med izdelavo diplomskega dela. Hvala mami za podporo in toleriranje dolgega študijskega obdobja.

Kazalo

Slike

Povzetek

Namen diplomskega dela je predstavitev prototipa sistema za optično preverjanje napak na lesu, konkretno na bambusovih deščicah, namenjenih izdelavi parketa. Za optično zaznavanje napak smo uporabili barvne kamere The Imaging Source DFK 23G018, ki so z računalniškim sistemom povezane preko Ethernet kabla. Predstavljene so štiri metode za obdelavo posnetih slik. Prva metoda izmeri dimenzije in določi robove opazovanega objekta oziroma lesene deščice. Znotraj določenih robov delujejo metoda za iskanje lukenj na objektu, metoda za iskanje poškodb na objektu in pa metoda za določanje barvne skupine objekta. Za simuliranje pomikanja deščic vzdolž procesne linije skrbi koračni servomotor. V okviru našega diplomskega dela je opisano tudi delovanje in upravljanje s krmilnikom motorja.

Ključne besede: optična kontrola, detekcija napak na lesu, računalniški vid, koračni motor.

Abstract

The objective of this thesis is to present a prototype of an optical wood control system, concretely, an optical measurement system for bamboo boards intended for parquet flooring. We used The Imaging Source DFK 23G018 color cameras for optical detection and measurement. In this thesis we present four methods for processing the images of wooden boards. The first method finds the edges of the boards and measures individual dimensions. By knowing the exact position of the boards in the images, the remaining three methods detect holes in the boards and different kinds of imperfections and sort the boards into appropriate color groups. To simulate the control part of the process line we used a step actuator. The functionality and the management of the actuator are also explained.

Keywords: optical measurement, wood imperfection detection, computer vision, step actuator.

Poglavje 1

Uvod

Raznovrstni kosi razrezanega lesa imajo številne značilnosti, ki jih človek lahko pregleda z očmi in zazna morebitne napake ter določi kvaliteto lesa. Pri avtomatizirani kontroli lesnih in drugih izdelkov pa je potrebno kontrolne sisteme naučiti prepoznavanja teh značilnosti, kar pa je zaradi njihovega velikega števila ena izmed največjih težav avtomatiziranih kontrolnih procesov.

Povpraševanje po avtomatizaciji pregledovanja kakovosti izdelkov in polizdelkov v lesni industriji zadnja leta strmo narašča. Faktor človeške napake je zelo velik v primerjavi s sistemi, podobnimi opisanemu v diplomski nalogi, hitrost in natančnost strojnega pregledovanja pa sta neprimerno višji. V času študija, predvsem v okviru študentskega dela in študijske prakse, sem v podjetju MBvision spoznaval problematiko s področja preverjanja kakovosti različnih izdelkov za lesno, kovinsko in avtomobilsko industrijo.

Zaradi vse večjega tržnega zanimanja za avtomatsko pregledovanje kakovosti daljših lesenih desk, smo se odločili, da za diplomsko nalogo izdelamo in opišemo delovanje prototipa stroja, ki optično pregleduje lesene deske in jih s pomočjo rotacijskega motorja ter valjev pomika po procesni liniji. Vzdolž procesne linije so nameščeni štirje moduli za optično analizo premikajočih se desk. Vsak modul je sestavljen iz ene barvne kamere in dveh luči za osvetlitev, nameščeni pa so tako, da objekt pregledajo od spodaj, od zgoraj in z obeh strani.

V okviru diplomske naloge je predstavljena simulacija oziroma prototip zgoraj opisanega sistema, sestavljata pa ga dva modula, ki objekt opazujeta od zgoraj in z leve strani glede na smer premikanja. Druga dva modula sta po komponentah in funkcionalnosti skoraj identična in za potrebe diplomske naloge nista obravnavana. Pomemben sestavni del sistema je tudi linijski servomotor, s pomočjo katerega je simulirano premikanje desk po liniji. Ker je sistem zasnovan tako, da je mogoče pregledovati deske različnih širin in debelin, je za dobro ostrino slike potrebno spreminjati fokus kamer ali pa kamere ustrezno približevati in oddaljevati. V končni verziji sistema bosta dva zgoraj omenjena linijska motorja služila za premikanje dveh kamer v zgornjem in v enem od stranskih modulov.

V nadaljnjih poglavjih je opisana celotna konstrukcija prototipa in vsa uporabljena strojna ter programska oprema. Sledi poglavje z natančnejšim opisom linijskega servomotorja in opis komunikacije krmilnika motorja z računalnikom. V petem poglavju je opisana programska koda strojnega vida z metodami razpoznavanja različnih napak na lesu in klasificiranja lesa glede na barvni odtenek. Sledi zaključek z rezultati in razmislekom o možnih izboljšavah sistema.

Glavni cilj diplomske naloge je predstavitev procesov za obvladovanje koračnega servomotorja, iskanje nepravilnosti na lesenih deščicah in preverjanje zmožnosti obdelave slik z večanjem hitrosti premikanja deščic po procesni liniji. Tipične nepravilnosti bambusovih deščic so prikazane na sliki 1.1.

Slika 1.1: Nepravilnosti bambusovih deščic

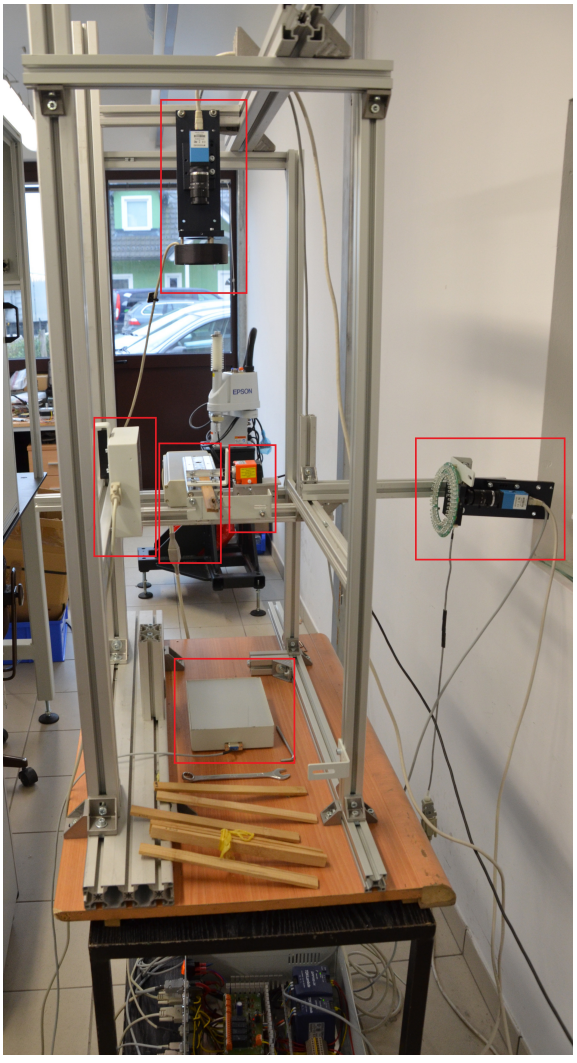
Poglavje 2

Konstrukcija prototipa

Konstrukcija prototipa sistema je fizično ločena na tri vertikalne in tri horizontalne nivoje. Prototip ima le dva modula s kamerami in lučmi, enega za pogled od zgoraj in drugega z leve strani. Na osrednjem nivoju je opazovani objekt, zgoraj ter z leve strani pa sta postavljeni kameri. Luči so postavljene v vseh štirih smereh, tako da objekt osvetlijo iz smeri posamezne kamere in njej nasprotne smeri. Ker se pogled in osvetlitev od zgoraj ter z leve strani ne motita, je objekt opazovan na istem delu procesne linije. V končni verziji sistema bosta spodnji in desni modul pomaknjena bodisi nazaj ali pa naprej po procesni liniji, saj se nasprotno ležeči moduli med seboj ovirajo pri zajemanju slike objekta.

Osnovo ogrodja in najnižji nivo predstavlja lesena plošča na dnu konstrukcije, na katero so privite štiri kovinske palice, ki štrlijo 130 cm navpično navzgor. Na osnovno leseno ploščo je pritrjena luč kvadratne oblike. Na višini 50 cm, kjer se nahaja drugi vertikalni nivo, so na pokončne palice privite štiri kovinske palice, ki tvorijo pravokotnik vzporedno z osnovno ploščo na dnu. Na ta pravokotnik je privit linijski motor z dolžino grede 25 cm, na motor pa je pritrjena 22 cm dolga bambusova deščica. Ta kratka deščica služi kot nadomestilo daljšim bambusovim deskam in je primerna za potrebe testiranja. Tekom izdelovanja diplomske naloge in testiranja programske kode je bilo zamenjanih več testnih palic z raznovrstnimi napakami, poškodbami

in različnimi barvnimi odtenki. S tem smo poskušali testirati kar največ možnih deščic z različnimi karakteristikami, da v končni verziji kontrolne faze proizvodnje ne bi prišlo do prevelikih odstopanj.



Slika 2.1: Konstrukcija prototipa

Glede na smer premikanja deščice, je na levi strani drugega nivoja na kovinsko palico pritrjena kamera z oddaljenostjo 45 cm od opazovanega objekta. Na skrajnem koncu kamere je privita luč v obliki krožnega kolobarja, ki objema objektiv, tako da ima kamera nemoten a vseeno osvetljen pogled na

objekt. Nasproti kameri in v enaki oddaljenosti od objekta je privita kvadratna luč, ki sveti nasproti kameri. Tako je na sliki kamere objekt odet v belo svetlobo, kar omogoča natančno izmero dimenzij in detekcijo morebitnih lukenj v objektu. Na zgornjem nivoju, 95 cm višje od spodnjega nivoja, se nahaja modul za pogled od zgoraj. Od modula s pogledom z leve strani se razlikuje le v smeri pogleda na objekt.

Poglavje 3

Strojna in programska oprema

Upoštevajoč letnico 2015, je v našem testnem projektu uporabljena strojna in programska oprema v večini zastarela. To velja predvsem za komunikacijske vmesnike. Razlog je razlika v ceni med sodobnejšimi napravami in starejšimi, že preizkušenimi ter še vedno dovolj zmogljivimi tehnologijami. Programska koda je pisana v jeziku C++, saj je, vsaj po naših izkušnjah, največ knjižnic in funkcij za strojni vid napisanih v tem jeziku.

3.1 Strojna oprema

3.1.1 Računalnik

Centralna procesna enota v računalniku je Intel i7 z oznako 4470K, velikost pomnilnika je 4 GB, disk pa je SSD velikosti 120 GB. S to konfiguracijo smo želeli zagotoviti nemoteno delovanje celotnega programa za pregledovanje slik lesenih deščic. Eden od glavnih ciljev diplomske naloge je bil ugotoviti, kakšna je najvišja hitrost premikanja lesenih desk po liniji, katerih slike bi bila procesor in pomnilnik še sposobna obdelati v realnem času.

Računalniku so dodane naslednje razširitvene kartice:

- Dve mrežni kartici za komunikacijo s kamerami na PCI-E vodilu.
- Kartica s paralelnimi vrati na PCI vodilu.
- Po meri narejena kartica na PCI vodilu, last podjetja MBvision z 9-pinskim priključkom za komunikacijo z elektronskim merilnikom razdalje

3.1.2 Konstrukcija

Industrijski barvni kameri DFK 23G618 podjetja The Imaging Source sta izbrani glede na razmerje med kakovostjo in ceno. Z ločljivostjo 640 x 480 točk (angl. pixels) zajemata slike 24-bitnega barvnega formata s frekvenco do 120 Hz. Objektiv podjetja Computar z velikostjo leče 50 mm omogoča oster pogled z razdalje 45 cm od objekta.

Slika 3.1: Barvna kamera DFK23G618

SMC linijski motor z gredjo dolžine 15 cm je koračni servomotor s krmilnikom. Krmilnik ima v EEPROM pomnilniku shranjenih 64 možnih korakov motorja, ki določajo pozicijo, hitrost, silo idr. Serijska povezava s pretvornikom med USB in RS485 do krmilnika motorja skrbi za komunikacijo med motorjem in računalniškim sistemom. Delovanje motorja in komunikacija s krmilnikom sta podrobneje opisana v naslednjem poglavju.

Slika 3.2: Koračni motor

Elektronski merilnik razdalje ima pomembno vlogo pri zajemanju slik s kamerami. Merilnik ima 2 m dolgo merilno žico in meri razdalje z natančnostjo do 0.15 mm. Četudi bi lahko trenutno pozicijo objekta pridobivali s podatki o stanju motorja, je elektronski merilnik zanesljivejši, njegova komunikacija z računalnikom pa hitrejša od tiste z motorjem.

Slika 3.3: Elektronski merilnik

Luči in poznavanje zakonov osvetlitve so pri optičnem zaznavanju napak velikega pomena, saj lahko izbira luči in njihova pravilna postavitev odločilno vplivata na kakovost zajetih slik. V konstrukciji sta uporabljeni dve vrsti luči glede na obliko, in sicer ena v obliki krožnega kolobarja, druga pa pravokotne oblike. Prva različica luči je pritrjena na konec objektiva kamere in sveti v smeri pogleda kamere. Sestavlja jo 24 LED diod, ki so enakomerno razporejene po kolobarju. Pravokotna luč sveti v nasprotni smeri, sestavlja pa jo 96 LED diod, razporejenih v pravokotnik. Pokrov iz rahlo zatemnjenega pleksi stekla skrbi za enakomerno razpršitev svetlobe.

3.1.3 Elektronika

Zaradi varnosti in nadzora nad napravami je med konstrukcijo in računalnikom škatla z elektroniko, ki skrbi za napajanje, proženje, prižiganje in ugašanje naprav. Na sprednji plošči škatle je nameščenih 13 priključkov, od tega 2 za LPT vrata, 4 za luči, 2 za kameri in priključek za napajanje motorja. Za morebitne potrebe po dodatnih kamerah ali laserjih so na voljo še štirje 9-pinski priključki. Glavno stikalo z varovalko omogoča nadzor nad dovajanjem elektrike vsem napravam v konstrukciji. Zaradi možnosti napake, ki bi lahko ustavila ali onemogočila delovanje glavnega programa ali pa celo računalnika, je v škatlo z elektroniko dodan čuvaj (angl. watch dog), v katerega program med delovanjem na vsakih 20 milisekund pošilja signal in tako sporoča, da je z delovanjem vse v redu. Če tega signala čuvaj ne dobi, onemogoči napajanje vsem napravam v konstrukciji. Pomemben del elektronike sta tudi dve kartici z optičnimi releji, ki skrbita za dovajanje napetosti lučem s karseda nizko latenco.

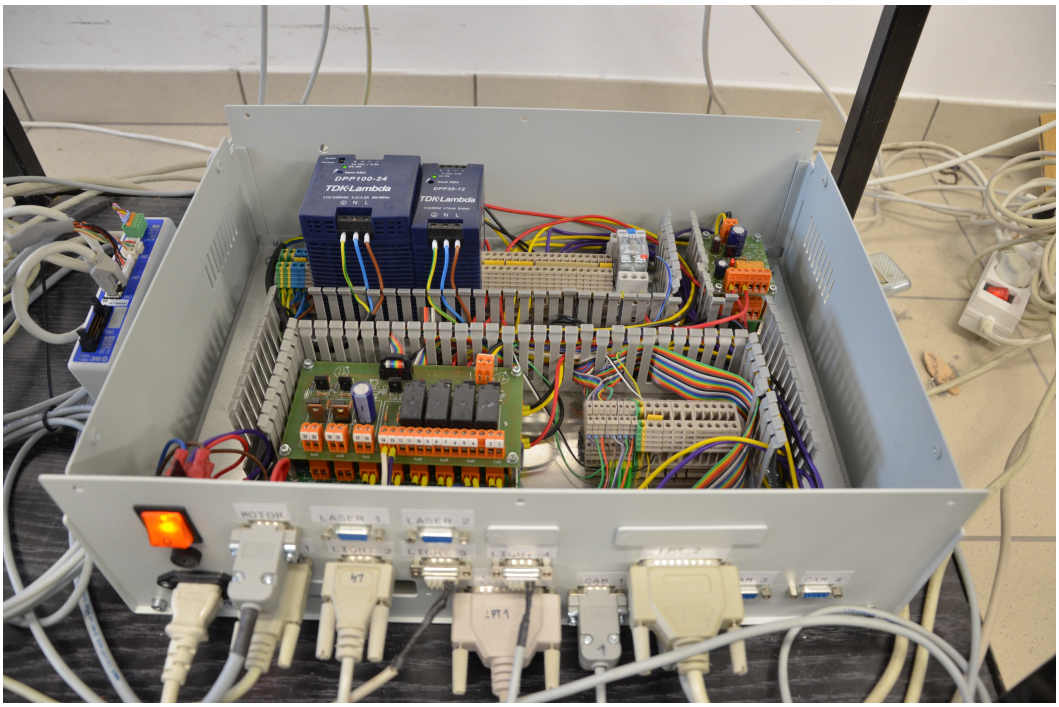
3.2 Programska oprema

3.2.1 Operacijski sistem in delovno okolje

Operacijski sistem računalnika je 32-bitna različica Windows 7, saj vsi ostali programi in knjižnice na tem sistemu preverjeno delujejo. Zaradi domačnosti je bilo izbrano Microsoftovo programsko okolje Visual Studio 2012, projekt pa je zasnovan kot MFC aplikacija.

3.2.2 Programske knjižnice

Pri pisanju programske kode smo si pomagali z različnimi programskimi knjižnicami. Še posebej velja izpostaviti knjižnico podjetja The Imaging Source s funkcijami za uporabo in nastavitev kamer, knjižnico wsc32 podjetja MarshallSoft Computing za enostavnejšo komunikacijo s krmilnikom

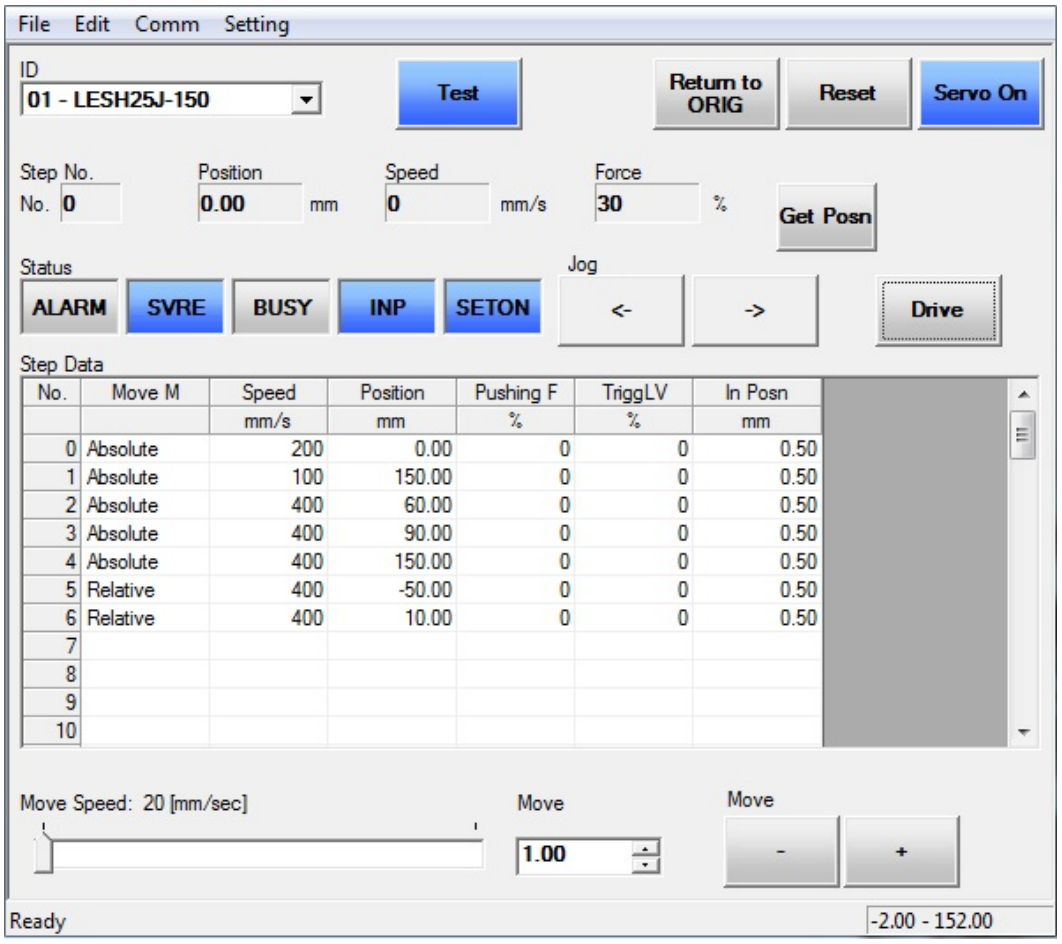


Slika 3.4: Škatla z elektroniko

motorja in knjižnico TvicHW32 podjetja EnTech, uporabljeno pri pošiljanju podatkov preko LPT vrat.

3.2.3 Upravljanje motorja

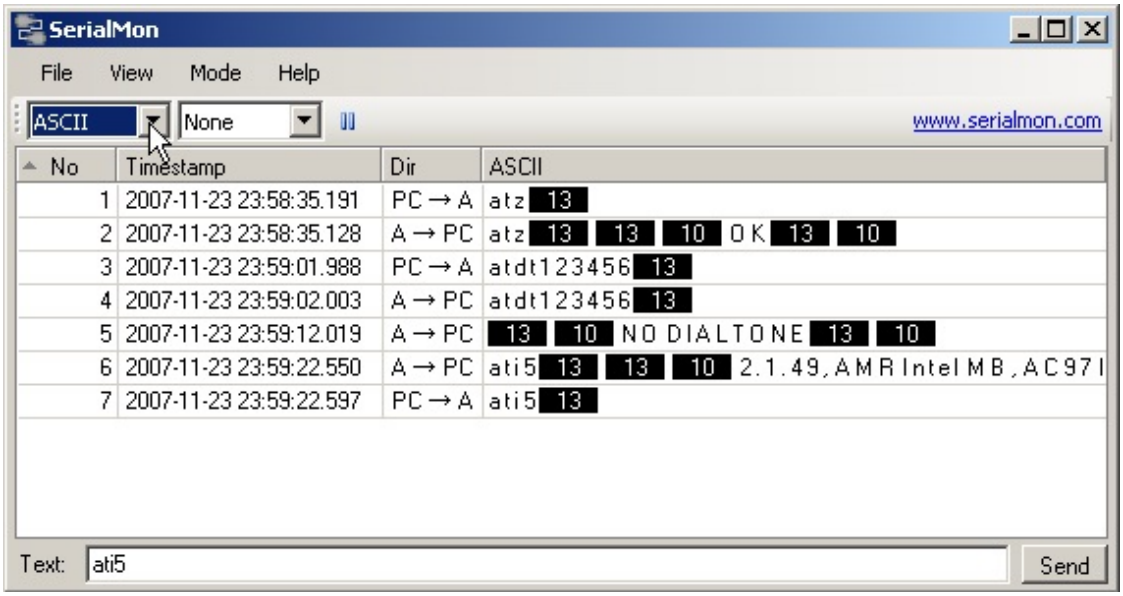
Učenje upravljanja SMC linijskega motorja je potekalo preko ACT Controller aplikacije podjetja SMC, preko katere smo dognali, na kakšen način motor deluje in kakšne so njegove zmožnosti. Slika 3.5 prikazuje osnovno različico programa, s katero lahko nastavljamo posamezne korake motorja, premikamo motor naprej in nazaj za določeno dolžino in z določeno hitrostjo.



Slika 3.5: Aplikacija za upravljanje motorja ACT Controller

3.2.4 Nadziranje serijske povezave

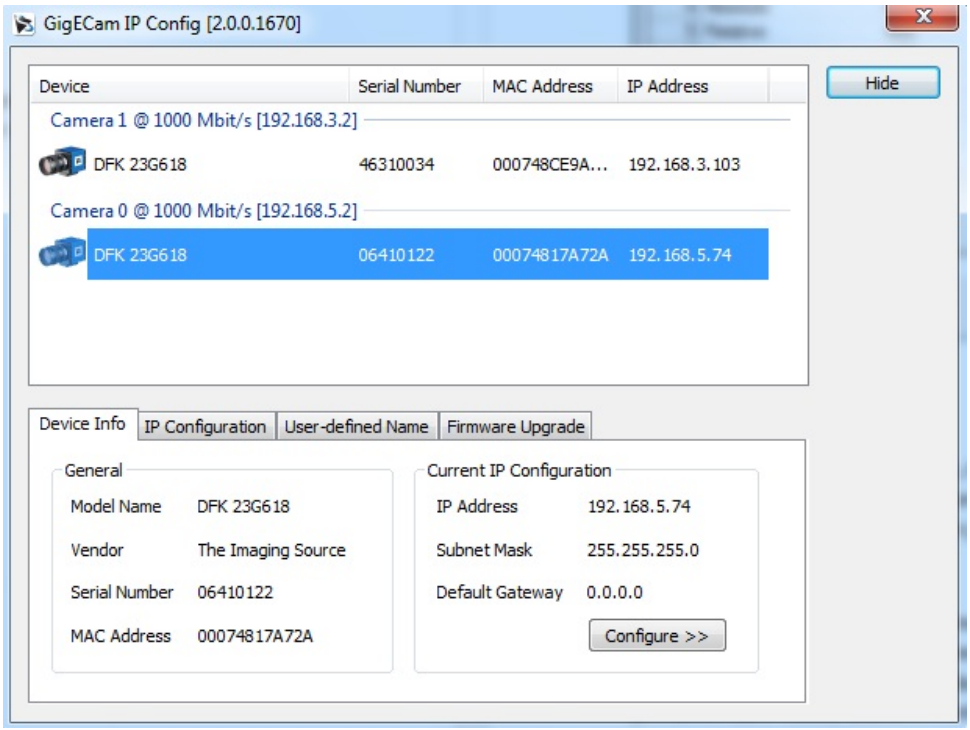
Za pridobivanje informacij o ukazih, ki jih ACT Controller aplikacija pošilja motorju, smo uporabili program SerialMon, ki nadzira promet preko serijske povezave na določenih kanalih in podatke o ukazih izpisuje na zaslon. Več o tem sledi v naslednjem poglavju.



Slika 3.6: SerialMon za nadzor serijske povezave

3.2.5 Gonilniki in nadzor nad kamerami

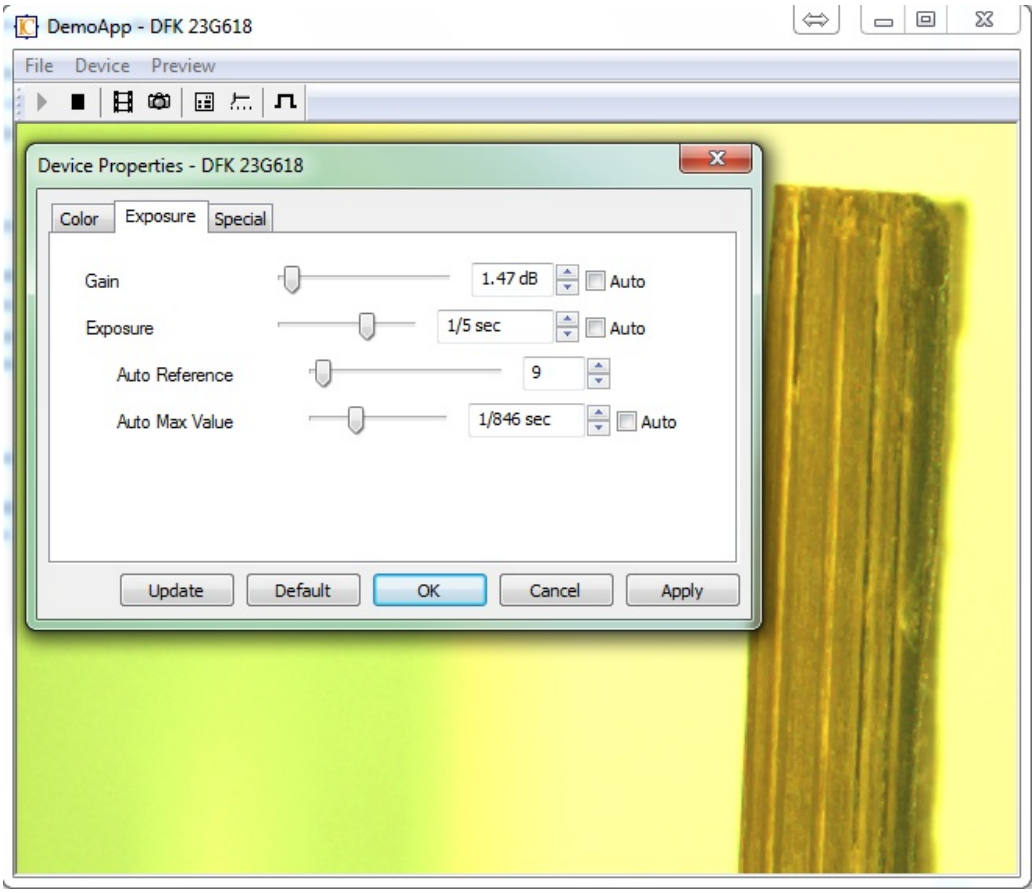
S pomočjo GigE programske opreme nadziramo delovanje večih kamer v sistemu. Ker je vsaka kamera z računalnikom povezana preko lastne omrežne povezave, je potrebno nastaviti posamezne IP naslove glede na omrežja. Z označitvijo ene izmed kamer se odprejo dodatne možnosti za vpogled v lastnosti in nastavljanje parametrov.



Slika 3.7: Upravljanje kamer s programsko opremo GigE

3.2.6 The Imaging Source Demo Application

Za lažje razumevanje delovanja kamer smo uporabljali The Imaging Source Demo Application, ki ponuja vse osnovne nastavitve kamer. Prikazuje lahko tudi živo sliko, s pomočjo katere smo si pomagali pri nastavljanju ostrine slike, ekspozicije, ojačanja svetlobe (angl. gain) in nastavljanju jakosti treh primarnih barv: rdeče, zelene in modre.



Slika 3.8: The Imaging Source Demo Application

Poglavje 4

Komunikacija

4.1 Komunikacija med krmilnikom motorja in računalnikom

Krmilnik koračnega servomotorja z računalnikom komunicira po protokolu, ki je podoben protokolu Modbus RTU. Prenosna linija ustreza standardu RS485 in deluje v načinu half-duplex. Sinhronizacijska metoda je start-stop, velikost znaka pa 8 bitov. Paritete ni. Dovoljena pasovna širina je med 9600 in 230400 biti na sekundo. Pomnilnik krmilnika je tipa EEPROM, kamor se shranjuje 64 korakov motorja in druge nastavitve. V posameznem zapisu koraka v pomnilniku so podatki o lokaciji, hitrosti, moči potiska in pospešku. Ko motorju pošljemo ukaz za premik na določeno pozicijo oziroma določen korak, se pri premiku upoštevajo vsi prej omenjeni podatki.

Nastavitev vseh parametrov posameznih korakov je s programom ACT Controller zelo enostavna in premikanje motorja na posamezne korake je ob sledenju navodilom proizvajalca razmeroma enostavno opravilo. Za delovanje našega sistema pa bi potrebovali nenehno spreminjanje pozicije motorja in ne vedno na ista mesta. Krmilnik uporablja EEPROM vrsto pomnilnika z omejenim številom branj in pisanj, zato se je pojavila potreba po dinamičnem spreminjanju pozicije motorja. Čeprav program ACT Controller omogoča premike na določeno razdaljo z določeno hitrostjo, brez da bi spremenil ka-

kega od korakov, primera ukaza v navodilih proizvajalca ni bilo moč najti in tudi po posvetovanju z inženirji podjetja SMC, nam ni kazalo nič bolje.

Slika 4.1: Krmilnik motorja LESH25RJ

S programom za nadzor določenega serijskega kanala SerialMon ter razbiranjem poslanih in sprejetih znakov med programom ACT Controller in krmilnikom motorja, nam je uspelo izluščiti serije znakov, ki pomenijo premik na lokacijo, ki ni zapisana med 64-timi koraki. Od septembra 2014 dalje je v uradnih navodilih za krmiljenje motorja tudi zgoraj omenjeni postopek. Čemu je bil do takrat skrit, se sprašujemo še danes.

Osnovna oblika ukaza je prikazana v spodnji tabeli:

Št. Naprave	Funkcija	Podatki	CRC
1 bajt	1 bajt	N bajtov	2 bajta

Prvi stolpec predstavlja identifikacijsko številko naprave, saj je naprav na eni povezavi lahko več. V drugem stolpcu povemo, kakšno funkcijo opravlja ukaz. Primeri funkcij so: branje, pisanje in preverjanje stanja motorja. Podatki v tretjem stolpcu se nanašajo na funkcijo in lahko spreminjajo ali berejo podatke npr. o korakih ali pa trenutnem stanju motorja in podobno. Podatkovni del ukaza je dolg največ 256 bajtov.

CRC preverjanje serije znakov po načinu Modbus16 na koncu ukaza skrbi za pravilnost poslanega niza znakov. Znaki so zapisani v šestnajstiškem sistemu.

4.2 Vzpostavitev komunikacije s krmilnikom motorja

Za vzpostavitev serijske povezave med krmilnikom motorja in računalnikom uporabljamo funkcije knjižnice WSC32 podjetja MarshallSoft Computing. SioReset funkcija odpre vrata serijske povezave, katerih zaporedna številka je prvi argument funkcije, in po njej pošlje ter nato sprejme testni niz znakov velikosti drugega in tretjega argumenta funkcije. Funkcija SioBaud določi pasovno širino na povezavi. Vsebina tabele start, ki vsebuje šestnajstiške zapise vrednosti znakov, se po povezavi pošlje s funkcijo SioPuts. V funkciji SioPuts je prvi argument št. od vrat povezave, drugi argument je niz znakov, ki ga želimo poslati, in tretji argument število poslanih bajtov v sporočilu. Funkcija SioGets prejme odgovor motorja in tako preveri, če je motor sprejel ukaz.

Koda 4.1: Vzpostavitev komunikacije s krmilnikom

```
void MotorSMC::EnableMotorCommunication(void)
{
code = SioReset(7, 1024, 1024);
code = SioBaud(7, 38400);

start[0]=01;
start[1]=05;
start[2]=00;
start[3]=0x30;
start[4]=0xff;
start[5]=00;
start[6]=0x8c;
start[7]=0x35;
SioPuts(7, start, 8);
SioGets(7, start, 8);
}
```

4.3 Primer ukaza za premik motorja

Primer ukaza, ki motor premakne na želeno pozicijo z določeno hitrostjo, potiskom in pospeševanjem:

01	10	91 02 00 10 20 (1.) 00 01 (2.) 01 F4 (3.) 00 00 3A 98 (4.)	72 48
		13 88 (5.) 13 88 (6.) 00 00 (7.) 00 00 (8.) 00 14 (9.) 00 64 (10.)	
		00 00 00 00 (11.) 00 00 00 00 (12.) 00 00 00 64 (13.)	

Vsak člen ukaza je zapisan šestnajstiško. Prvi stolpec pove, da komuniciramo z napravo pod številko 1. Drugi stolpec pove, da gre za ukaz za pisanje. V tretjem stolpcu so podatki o pisanju, in sicer:

1. Izbrana pomnilniška lokacija, kamor se bo pisalo, je 91 02. Na tej lokaciji se rezervira prostor za 16 (10) besed in vanjo zapiše 32 (20) znakov.
2. Način premika 00 01 pomeni absolutno. Drugi način je relativno glede na trenutno pozicijo.
3. Hitrost premika je 01 F4, kar pomeni 500 mm na sekundo.
4. Pozicija 00 00 3A 98 pomeni 150 milimetrov (absolutno).
5. Pospešek 13 88 znaša 5000 mm/s^2.
6. Pojemek 13 88 znaša 5000 mm/s^2, tako da se pospešek in pojemek odštejeta.
7. Sila potiska znaša 0
8. Sprožitveni prag znaša 0
9. Hitrost potiska 00 14 pomeni 20 mm na sekundo.
10. Gonilna sila 00 64 pomeni 100

11. Izhodno območje 1 je 0.

12. Izhodno območje 2 je 0.

13. Motor se ustavi, ko pride v območje +- 100 μm (00 00 00 64) od pozicije, določene v točki 4.

V četrtem stolpcu sta vrednosti CRC pregleda celotnega niza znakov, vendar je njun vrstni red je zamenjan.

4.4 Stanje motorja

Glede na vsa možna stanja, v katerih se motor lahko nahaja, je potrebno napisati programsko kodo, ki se na ta stanja ustrezno odzove.
Ukaz za preverjanje trenutnega stanja motorja:

01	02	00 40 00 10	78 12

V programski kodi, namenjeni obvladovanju motorja, točneje v časovnem števcu (angl. timer), se stanje motorja preverja vsakih 250 milisekund. Krmilnik motorja vrne šestnajstiško vrednost, ki jo v programu spremenimo v desetiško »kodo«.
Pri preverjanju stanja smo pozorni na naslednje kode:

- 0 – motor se ne odziva, kar pomeni, da ali ni vključen ali pa je nekaj narobe s komunikacijo. Reakcija je poskus vklopa motorja.

- 2 – motor nima določene referenčne (začetne) točke. Reakcija je ukaz za premik motorja na prvotno pozicijo (angl. return to original position).

- 7 – motor je v fazi premikanja. Reakcija ni potrebna. Počakamo, da motor pride na cilj.

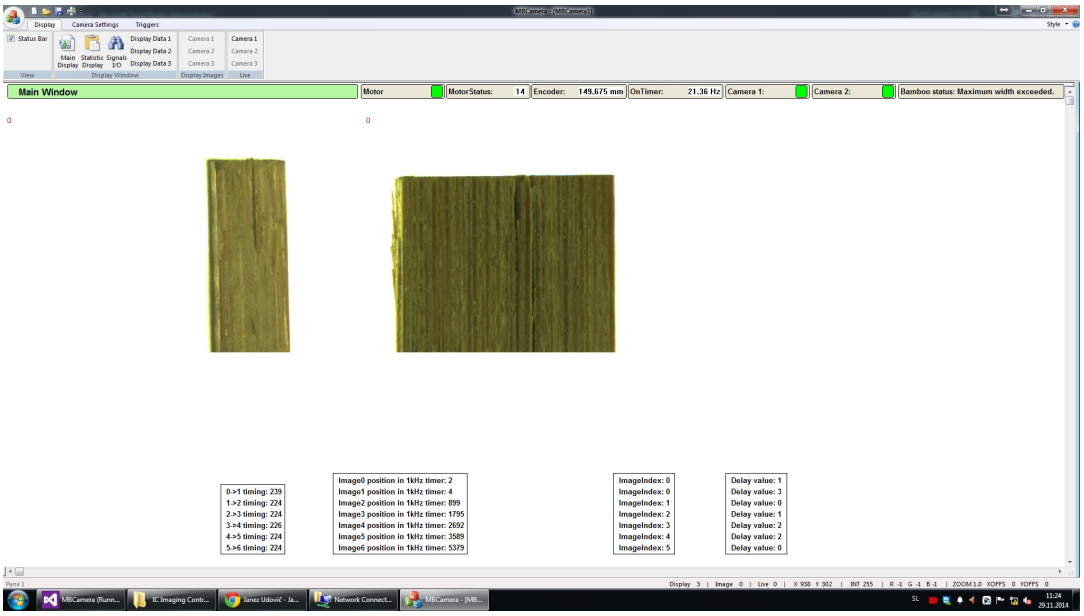
- 8 ali 12 – motor ima neznano napako, ki je ne znamo ponoviti. Reakcija je ponoven zagon motorja.
- 142 – motor stoji, ampak izven pozicije na katero je bil poslan. Reakcija je ponastavitev pogona motorja.

Poglavje 5

Delovanje programa

Program je napisan v programskem jeziku C++, projekt v Visual Studiu pa nastavljen kot MFC aplikacija, ki ponuja enostavno delo z meniji, kar je glavni razlog za njegovo izbiro. Aplikacija sestoji iz več zaslonov. Na sliki 5.1 vidimo osnovni zaslon, ki se pojavi ob zagonu aplikacije in nam prikazuje sliki opazovanega objekta od zgoraj in z leve strani. Spodaj pod slikama deščic je z namenom sprotnega pregleda skrbnika sistema prikazanih več spremenljivk, ki skrbijo za pravilno delovanje sistema.

S pomočjo menija na vrhu se je mogoče premikati med zasloni, ki imajo vsak svojo funkcijo. Vsaka kamera si lasti svoj zaslon, kjer je mogoče prikazati živo sliko. Na enem izmed zaslonov so npr. predstavljeni parametri, kot so hitrost pomikanja motorja in nastavitve kamer, ki jih operater oziroma nadzornik sistema lahko spreminja.



Slika 5.1: Glavn zaslon programa

5.1 Inicializacija in nastavitev kamer

Po začetni definiciji dveh objektov razreda CImagingSource, ki predstavljata kameri, je potrebno vsako kamero inicializirati. Pri tem je potrebno paziti na vrstni red kamer in na njuni imeni, kot je to določeno v programu The Imaging Source Demo Application. Podati je potrebno tudi format zajemanja slike, globino posamezne točke (angl. pixel) in ločljivost. V našem primeru gre za barvno sliko s točkami globine 3, kjer rdeča, zelena in modra barvna komponenta zasedajo po 8 bitov – od tu RGB24. Velikost slike je nastavljena na največjo vrednost 640 x 480 točk. Funkcija Start nastavi število slik na sekundo (angl. frames per second) in postavi zastavice sprožilca in poslušalca (angl. listener).

Z uporabo funkcij SetGainAbsolute, SetExposureAbsolute in SetWhiteBalance se nastavlja svetlost slike, ekspozicijo in intenziteto posameznih barv. Čeprav teorija o nastavljanju prej omenjenih lastnosti ni zapletena, je v praksi določanje optimalnih nastavitev v večini primerov plod preizkušanja

različnih vrednosti, dokler zajeta slika ne ustreza pogojem za obdelavo. Velik faktor pri tem je svetloba iz okolice, zato so v končnih izvedbah kontrolnih sistemov posamezni moduli s kamerami in lučmi svetlobno izolirani.

Koda 5.1: Nastavitev lastnosti kamer

```
if (cam[0].Init(0,_T("DFK 23G618"),0,_T("RGB24 (640x480)") == 1)
{
        cam[0].Start(120, true, true);
        cam[1].SetGainAbsolute(54);
        cam[0].SetExposureAbsolute(1/2500.0);
        cam[0].SetWhiteBalanceAbsolute(90, 70, 90);
}
if (cam[1].Init(1,_T("DFK 23G618 1"),0,_T("RGB24 (640x480)")) == 1)
{
        cam[1].Start(120, true, true);
        cam[1].SetGainAbsolute(60);
        cam[1].SetExposureAbsolute(1/2677.0);
        cam[1].SetWhiteBalanceAbsolute(90, 75, 90);
}
```

5.2 Usklajevanje motorja in električnega merilca razdalje

Da je začetna pozicija motorja enaka vrednosti 0 na merilcu razdalje, je potrebno ti dve napravi ob zagonu programa uskladiti. Košček kode spodaj je del avtomata, ki vsako milisekundo preverja stanje merilca (v kodi Encoder). Spremenljivka »motorInPosition« pove, ali sta motor in merilec usklajena. Če nista usklajena, se preveri stanje motorja in njegovo pozicijo. Ko je motor v začetni poziciji, se trenutni odmik merilca shrani in vsakokrat odšteje od dejanske vrednosti merilca. V nasprotnem primeru motorju pošljemo ukaz za premik na začetno pozicijo. Takrat dobi spremenljivka »motorInPosi-

tion« vrednost »true«, kar pomeni, da sta motor in merilec usklajena in postopka preverjanja usklajenosti ne ponavljamo več.

Koda 5.2: Usklajevanje motorja z merilcem razdalje

```
encoderValue = pci->GetDataByteEncoder(); //get encoder value
encoderValue = encoderValue - pciEncoderOffset;

if (motorInPosition == false)
{
        if ((motor.check == MOTOR_OK) || (motor.check ==
            MOTOR_NEEDS_CALIBRATION))
        {
                motor.ReadPosition();
                //motor is in start position and has reference point
                    set
                if ((motor.position == 0) && (motor.check != 2))
                {
                        pciEncoderOffset = pci->GetDataByteEncoder();
                        motorInPosition = true;
                }
                else
                {
                        motor.ReturnToOriginalPosition();
                }
        }
}
```

5.3 Zajemanje slik

Najdaljši premik linijskega motorja znaša 15 cm in ta dolžina je približno 7-krat večja od vidnega polja kamere. Dolžina vidnega polja nam določa, kdaj je potrebno sprožiti kamero. Premikanje motorja je, kljub nastavitvi brez pospeška, neenakomerno, zaradi česar se je pojavila potreba po elektronskem merilcu razdalje. Ta nam v trenutku zajetja slike sporoči točno razdaljo, kar je vidno v Koda 5.3.

Koda 5.3: Zajem slike glede na pozicijo motorja

```
if (takePicture == 1)
{
        if (encoderValue > encoderDistance)
        {
                distance[distanceCounter] = cam[0].imageIndex;
                imageEncoderTable[distanceCounter] = encoderValue;
                lpt[0]->SetDataBits(0x0F, 1);
                lpt[1]->SetDataBits(0x03, 1);
                imageTiming[timing] = timerInterval -
                    oldTimerInterval;
                timing++;
                imageAdjustment[adjustmentIndex] = encoderValue -
                    encoderDistance;
                encoderDistance = encoderValue + fieldVision;
                distanceCounter++;
                adjustmentIndex++;
                oldTimerInterval = timerInterval;
        }
        else
        {
                lpt[1]->SetDataBits(0x03, 0);
                lpt[0]->SetDataBits(0x0F, 0);
        }
}
```

Tudi ta koda je del avtomata, ki vsako milisekundo preverja, če so pogoji za proženje kamere izpolnjeni. Ko vrednost merilca preseže določeno mejo, se sprožijo vse štiri luči in obe kameri.

Funkcija SetDataBits preko prvih LPT vrat pošlje signal za prižiganje vseh štirih (0F) luči, preko drugih LPT vrat pa pošlje signal za proženje obeh (03) kamer.

Meja je določena kot vsota trenutne vrednosti merilca in spremenljivke »fieldVision«. »fieldVision« je določena kot vsota vrednosti vidnega polja in rezervnega števila. Rezervno število služi kot varovalka, katere vrednost je določena na podlagi testov neenakomernega premikanja motorja. Z dovolj velikim rezervnim številom se prepričamo, da motor v trenutku zajemanja slike ni bil prehiter, saj bi v tem primeru kamera posnela sliko prepozno in bi del deščice lahko izpustili. Tako pa vsaka slika, razen prve, vsebuje del prejšnje slike in to nam zagotovi, da je deščica v celoti pregledana. Medtem se v posebne tabele shranjujejo vrednosti merilnika, s pomočjo katerih dobljene slike ustrezno odrežemo in s tem zmanjšamo čas obdelave.

5.4 Detekcija stranskih robov

Pomen luči, ki sveti nasproti smeri kamere, je ustvariti zelo svetlo oziroma belo ozadje, kar izrazito poudari kontrast med deščico ter ozadjem na sliki in tako omogoča zanesljivejšo detekcijo robov. Del programske kode za iskanje in izris obeh robov je prikazan v Koda 5.4.

Koda 5.4: Detekcija stranskih robov

```
bamboo.rect.SetRect(x1, y1, x2, y2);
bamboo.HorizontalIntensity(image, 2);
bamboo.DetectTransitionsWhiteToBlack(whiteThreshold,
    blackThreshold);

for (y = i + y1; y < image.height - y2; y++)
{
        if (bamboo.detectedPointsLightToDark.size() > 0)
        {
                leftEdge[y] = (int)
                    bamboo.detectedPointsLightToDark[0].x;
                pDC->SetPixel(leftEdge[y], y, RGB(255,0,0));
        }
}
```

Po uspešnem določanju dveh skrajnih točk, ki predstavljata najbolj levi in najbolj desni zgornji konec objekta na sliki, se blizu ordinatam teh točk določijo pravokotniki, po katerih se z uporabo funkcije DetectTransitionsWhiteToBlack (in DetectTransitionsBlackToWhite za desni rob) nadaljuje iskanje drastične spremembe povprečne intenzitete točk. Na mestu teh sprememb se določi rob opazovanega objekta in izriše rdeča oziroma modra točka. Risanje točk s funkcijo SetPixel je zamudna operacija, ki nam omogoči preverjanje pravilnosti delovanja, zato je uporabljana le v fazi testiranja. Na podlagi točk, ki označujejo levi in desni rob, se nato izračuna širino in preveri dimenzijsko ustreznost objekta glede na podane tolerančne vrednosti.

Intenziteta pri barvni sliki je povprečje vrednosti rdeče, zelene in modre barve posamezne točke, določene z 8 biti. Če je slika sivinska, ima globino ena. To pomeni, da intenziteto vsake točke predstavlja 8 bitov, kjer 0 pomeni popolnoma črno in 255 popolnoma belo barvo.

Slika 5.2: 24 bitni zapis točke

5.5 Detekcija nepravilnosti na površini opazovanega objekta

Z namenom pohitritve delovanja algoritma za detekcijo lukenj, razpok in ostalih poškodb na opazovanem objektu smo uporabili podatke iz algoritma za detekcijo robov, ki določijo področje našega zanimanja, preostalih delov slike pa pri pregledu ne upoštevamo.

Koda 5.5: Iskanje napak na površini objekta

```
for (int j = focusLeft; j < focusRight; j++)
{
        for (int i = focusTop; i < focusBottom; i++)
        {
                position2 = (blackDots.height-j-1) * blackDots.width
                    + i;
                blackDots.buffer[position2] =
                    bigImg2->GetAverageIntensity(i + yOffset, j +
                    xOffset);
        }
}
```

```
blackDots.FindCommonPointsBlack(40, 185, 15);
blackDots.DrawPoints(&dc);
```

Koda 5.5 prikazuje dve zanki z mejami, določenimi z robovi pravokotnika, s katerim omejimo področje iskanja napak na sliki. Funkcija FindCommonPoints ima za prvi parameter določen prag - intenziteto pik, ki jih klasificiramo kot črne. Drugi in tretji parameter funkcije FindCommonPoints sta določena kot maksimalno in minimalno število pik, ki se držijo ena druge in pomenijo napako. FindCommonPoints shrani podatke o pravokotnikih, ki obdajajo najdene točke. Tako je mogoče pridobiti podatke o dolžini in širini posamezne skupine črnih točk. Glede na zgoraj omenjene podatke o skupinah črnih točk je mogoče določiti zgornjo mejo, pri kateri je opazovan objekt še ustrezen. Včasih lahko kvaliteto objekta oziroma izdelka določa tudi odstotek seštetih skupin črnih točk glede na celotno površino ne glede na velikost oziroma majhnost posamezne skupine. V tem primeru površine vseh skupin črnih točk seštejemo in delimo s celotno površino opazovanega objekta. Za potrebe testiranja smo uporabili tudi funkcijo DrawPoints, ki zgoraj določene skupine točk pobarva z živimi barvami.

5.6 Povečanje izrazitosti napak

Intenziteta točk na sliki objekta, kjer je poškodba oziroma napaka, ima lahko podobne vrednosti kot deli površine objekta, ki ustrezajo kakovostnim standardom in niso obravnavani kot napake. Da bi dejanske napake poudarili in jih tako lažje ločili od npr. prahu ali pa letnic na lesu, smo uporabili dve metodi za filtriranje in primerjanje zanimivih območij na sliki objekta.

5.6.1 Filtriranje slik

Koda 5.6 prikazuje del funkcije za fitriranje slik objekta. Gre za počasno operacijo in jo zato uporabljamo le, ko problema ni več možno rešiti z osvetlitvijo ali optiko.

Koda 5.6: Postopek filtriranja slik

```
for (int j ) 640 - bottomFocus; j < 640 - topFocus - maskHeight /
    2; j++)
{
        for (int i = 1; i < width - maskWidth / 2; i++)
        {
                for (int y = -maskHeight / 2; y <= maskHeight / 2;
                    y++)
                {
                        for (int x = -maskWidth / 2; x <= maskWidth /
                            2; x++)
                        {
                                aroundIntensity[tempPosition] =
                                    pBuf[((j+y) * width + (i+x)];
                                tempPosition++;
                        }
                }
                for (int k = 0; k < tempPosition; k++)
                {
                        intensity += aroundIntensity[k];
```

```
                }
                intensity /= (int)tempPosition;
                if (intensity > 255)
                {
                        intensity = 255;
                }
                position = (j * width) + i;
                buffer[position] = intensity;
                tempPosition = 0;
        }
}
```

Dolžina in širina matrike, podani v argumentu funkcije, nam določata število točk okrog točke, ki je trenutno v obdelavi. Povprečna intenziteta vseh teh točk je zapisana v trenutno točko. Točke, ki so temnejše od ostale površine se še bolj potemnijo, hkrati pa se tudi svetlejše točke posvetlijo. S tem postopkom se poudari robove morebitnih napak na površini in omogoči natančnejše zaznavanje in merjenje napak.

5.6.2 Dodatno filtriranje in odštevanje slik

Za še bolj natančno zaznavanje poškodb na površini smo uporabili tudi funkcijo, ki kreira novo sliko z identično velikostjo in globino. Na tej novi sliki se izvede filtriranje s širšim ali daljšim filtrom od prvotnega filtra. Ti dve različno filtrirani sliki med seboj odštejemo, kar pomeni poiskati razliko v intenziteti med istoležnimi točkami obeh slik, in kot rezultat dobimo tretjo sliko s poudarjenimi robovi napak. Ta operacija je tudi do 10 krat počasnejša od osnovnega filtriranja in jo uporabljamo le, ko zaradi slabše osvetlitve ni več mogoče natančno definirati napak na površini.

5.7 Ločevanje deščic glede na barvni odtenek

Če je deščica po opravljeni kontroli označena kot dober izdelek, primeren za nadaljno uporabo, se na sliki opravi še postopek določanja barvnega razreda. Funkcija, ki določa barvni razred, izračuna povprečno intenziteto vseh točk na deščici. Ker bambusov les nima grč in je v primerjavi z drugimi vrstami lesa barvno enoličen, je postopek klasificiranja deščic po barvi zelo enostaven. Pri drugih vrstah lesa se lahko barva drastično spreminja glede na oddaljenost od središča debla, čemur lesarji pravijo črnina in belina. Na podlagi rezultatov zgoraj omenjene funkcije se na koncu kontrolnega postopka deščice razvrstijo v različne zaboje glede na barvni razred. Na sliki 5.3 so prikazani različni barvni odtenki bambusovih deščic.

Slika 5.3: Barvna raznolikost bambusovega lesa

Poglavje 6

Rezultati in zaključek

Pri načrtovanju sistema smo poskušali predvideti največje težave, s katerimi bi se najverjetneje srečali pri pregledovanju pomikajočih se deščic po proizvodni liniji. Eden izmed glavnih ciljev je bilo zagotoviti karseda hitro pomikanje deščic vzdolž proizvodne linije, ne da bi pri tem trpela natančnost pregledovanja. Z ustrezno postavitvijo luči smo poskrbeli za natančno in zanesljivo pregledovanje napak, z dodajanjem električnega merilca razdalje pa smo rešili problem nesinhroniziranega sestavljanja večih slik v eno samo sliko, na kateri poteka pregled. Zadani cilj pri načrtovanju sistema je bilo premikati deščice s hitrostjo 500 mm/s, pri natančnosti iskanja napak, ki so večje od 0.5 mm in s ponovljivostjo meritev, ki ne odstopajo za več kot 5 μm pri napakah oziroma poškodbah.

6.1 Meritve in odstopanja

Na začetku razvoja programskega dela sistema je bila hitrost pomikanja deščic 100 mm/s. Pri tej hitrosti je bila povprečna napaka zajemanja slik zanemarljiva, saj sta imeli kameri več kot dovolj časa za zajem in pošiljanje slik v obdelavo. Pomnilnik in procesor sta kljub velikim količinam podatkov o opazovanem objektu uspešno izvajala obdelavo, saj so slike prihajale razmeroma počasi. Pri 1000-kratni ponovitvi pregleda iste deščice je prišlo

do največ 0.2 mm odstopanja pri merjenju velikosti napak in širine oziroma debeline deščice. Pri dvokratnem povečanju hitrosti pomikanja deščic na 200 mm/s so bili rezultati meritev identični tistim s hitrostjo pomika 100 mm/s.

6.1.1 Meritve pri polni hitrosti pomikanja

Z namenom testiranja sistema pri hitrosti 500 mm/s smo morali dosedanji linijski motor zamenjati z novim modelom, ki to hitrost doseže in ima za nameček že 50 mm daljšo gred. Po 1000 pregledih iste deščice z novimi parametri sistema so bili rezultati pričakovano slabši. Odstopanja od prave vrednosti meritev so bila izven ranga 10 procentov glede na velikost napake, kar je bila približna tolerančna meja, zastavljena na začetku projekta.

Ker je 500 mm/s pričakovana minimalna hitrost pomikanja deščic, smo morali izboljšave iskati v optičnem delu sistema. Nakup novih, boljših kamer z večjo ločljivostjo bi pomenil prevelik strošek, hkrati bi se z večjimi slikami povečal tudi čas obdelave in velikost pomnilnika, potrebnega za nemoteno delovanje sistema. Rešitev problema smo našli v lečah, ki so nam omogočile večjo povečavo iz 50 na 75. Z večjo povečavo smo zaobšli potrebo po nakupu novih kamer; čas obdelave se je sicer na račun povečanja območja pregleda na slikah povečal sorazmerno z optično povečavo, kar pa ni ogrozilo tekočega delovanja sistema.

6.2 Prostor za izboljšave

V našem primeru je bila zadostna hitrost premikanja zelo nizka, saj se pri nekaterih proizvajalcih lesene deske vzdolž proizvodne linije premikajo tudi s hitrostjo 2 m/s in več. V takem primeru bi za tekoče delovanje kontrolnega sistema potrebovali veliko boljše kamere in močnejšo osvetlitev, saj bi zaradi hitrosti premikanja in s tem povezanimi problemi pri zajemanju slik potrebovali zelo nizko ekspozicijo kamer. Proti koncu razvijanja prototipa smo našli način, kako brez večjih posegov v strukturo programa zamenjati operacijski sistem iz 32-bitnega na 64-bitnega, kar je odprlo dodatne možnosti za nad-

gradnjo strojne opreme, predvsem pomnilnika, kar nam omogoča uporabo kamer z večjo ločljivostjo, če bi bilo v prihodnosti to kdaj potebno.

Vse naštete izboljšave pomenijo večji strošek izgradnje in razvoja takega sistema, kar lahko pripelje do nekonkurenčnosti na trgu kontrolnih sistemov v industriji.

6.3 Zaključek

Sistemi za kontrolo proizvodov v industriji postajajo skoraj tako pomembni kot sistemi za izdelavo proizvodov, saj reklamacije zaradi neustreznosti proizvodov lahko pomenijo izgubo posla. Na svetu obstaja veliko podjetij, ki ponujajo rešitve za kontrolo proizvodov z naprednejšimi tehnologijami tako v strojni opremi kot tudi v programski.

Namen te diplomske naloge je bil razviti konkurenčni sistem z opremo, ki si jo malo podjetje v sklopu razvoja in raziskav lahko privošči, in na osnovi pridobljenega znanja začeti z razvojem podobnih sistemov. Ta vizija se nam je začela uresničevati, saj je sistem, ki je na moč podoben temu iz diplomske naloge, ravnokar na testiranju v proizvodnji enega najuspešnejših slovenskih podjetij.

Literatura

[1] J. R. Parker, Algorithms for Image Processing and Computer Vision, John WIley and Sons, Inc., 2010

[2] Dr. Višnja Henč-Bartolić, Dr. Petar Kulišić, Valovi i optika, Školska knjiga, 1989

[3] Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, John Wiley and Sons Ltd, 2014